QADAAD ASSEMBLY PROGRAM Brief Description August, 1962

"Quick-And-Dirty Assembler And Documentor"

QADAAD is an assembly program for a Univac Solid State II (80 column) configuration with at least 5000 words of drum memory. It assembles from tape to tape at a speed of about 600 lines per minute. QADAAD provides facilities for hand minimization of latency, for relocatable output, and a special optional feature which draws flow charts of the program as an extra pass.

Card Format

cols 1-5	A field:	location
col 6	AR field:	relocation code for A
cols 7-9	AH field:	latency code for A
cols 10-12	OP field:	symbolic operation code
col 13	IR field:	index register or sign
cols 14-18	M field:	M-address
col 19	MR field:	relocation code for M
cols 20-22	MH field:	latency code for M
cols 23-27	C field:	C-address
col 28	CR field:	relocation code for C
cols 29-31	CH field:	latency code for C
cols 32-80	remarks.	

Symbolic Fields

The A, M, and C fields contain information of the following types:
(here ∧ indicates a blank column)

1. Regional addresses, Xnnnn where X is any non-blank character and n represents a digit, e.g. A0000. A0003 is three locations greater than A0000. A9999 is one location less than A0000.

2. Absolute addresses, ∧NNNN where N represents either a digit or one of the undigits A,B,C,F,G,H. This address is simply reproduced in

the output. Examples: 4078, B2AB, H66H.

3. Local addresses. In A these are of the form n∧∧∧∧, and in M or C they are of the form nF∧∧∧ or nB∧∧∧ (referring to the next or previous A of n∧∧∧∧ respectively).

4. "Self": *∧∧∧∧ refers to A address. If this address occurs in A or on a control operator card it refers to the previous A address.

5. Pair address, any address of the form &WWWW or -WWWW where W's are arbitrary, indicates two adjacent addresses as used in "c + 1 conditions." The - is the first of the pair, the & is the second or overflow address.

6. Blank address. In A field, indicates the address corresponding to a blank M or C address on the preceding instruction. In M if the address is ignored by the Op (e.g. ATL or HLT) this is equivalent to *. In C if the address is ignored (e.g. CLA) the C address is set equal to M. Otherwise the next A field must be blank and it refers to this next A address.

7. Symbolic address. This is any address other than the above, except the leftmost character must be nonblank and non-numeric. Several symbolic addresses are pre-defined: RA is 000A, RL is 000B, RX is 000C; RB4, RB5,..., RB9 are respectively B2AB, B3AB, B5AB, B6AB, B7AB, B8AB. If multiple assembly is used these predefined symbols disappear, however.

H Fields

The AH, MH, and CH fields are relevant only if the corresponding address is undefined, i.e. if it has not appeared earlier. When the address is undefined, the H field is used to control storage allocation as follows:

1. ∧∧∧ or D∧∧ Choose the best available place on the drum.
2. H∧∧ Choose the best available place on the high-speed bands.
3. C∧∧ Choose a place in the B001-B999 core storage area.
4. nnn Assign on this level on the drum.
5. ∧nn Assign on this level on the high-speed bands.
6. +nn Assign nn higher than the normal rules would say.

R Fields

The AR, MR, and CR fields are merely transferred to the output area and are used to prepare FORTRAN subprograms. The FORTRAN assignment ignores the 4-bit and then has the following code:

	0 absolute	5 external reference
	1 unique storage	6 in AR for ALF table entry
^	or 2 program storage	7 cross reference
	3 common storage	8 special program storage

IR Field

The IR field is normally used to specify index register modification. If blank there is no indexing, otherwise the digits 1-9 are used to indicate modification by RB1-RB9. (In the control operators CON NUM ZON and ALF which provide constants the IR field is used to denote the sign instead, and then the sign is 0 if IR is blank, the sign is 1-9 if the IR field is 1-9.)

A convenient literal constant feature is included, so that if the IR field contains the character # the M and C fields are filled with a ten-digit constant which is positive. This constant is assembled separately, then the M address of the instruction refers to this constant, and C address is treated as if it were blank.

Example: ADD#01234 56789

is equivalent to ADD CONST and the line

CONST CON 012340 567890

appearing later. (Notice that the MR and CR fields of the constant are set to 0; the AR field is set to the MR field of the instruction.) The MH and CH fields of the instruction retain their normal significance.

Warning

Do not attempt to assembly anything into location 0000; this location is sacred to QADAAD. As a matter of fact the loading routine supplied on the output tape goes into band 0000 so it is wise not to assemble into any of the locations 0000-0399.

OP Field

The operation code field is filled with either a three-letter mnemonic op-code as used in the S4 assembly system, or it is one of the control operators listed below. Operators which require a 4-bit in the sign digit are not provided.

AAA . This is used for comments only.

CPY. The card-to-tape pass uses this. M and C are absolute addresses which refer to line numbers on the previous assembly listing; lines M through C are copied from tape in place of this card.

FIN. This is the last card of a program, signalling the end for the card-to-tape pass.

END. This is the second-last card of a program, signalling the end of the program. It can also be used to separate programs in a multi-program assembly. The M address must be defined, and the loading routine will be set to halt and transfer to this address.

BLR and BLA. Reserve or unreserve drum locations starting at M and ending at C. M and C must be defined addresses on the drum. If the CH field is nonblank, take n to be CH mod 100, indicating that we reserve or unreserve every n-th word. If the CH field is blank it means take n to be 1. If A is nonblank an EQU operation is also performed (See EQU). If C is blank it is taken equal to M.

COR. Reserve core locations in the B001-B999 area. M must de defined. M locations in the core are reserved. If A is non-blank an EQU to the first address of this block is also performed (see EQU). For example, R0001 COR 0030 reserves thirty core locations as the locations R0001 through R0030.

EQU. A must be an undefined regional, forward local, or symbolic address. (Not a pair address.) It is then defined to equal the equivalent of M which must be defined.

HHH. All succeeding blank H-fields are overridden by the contents of the MH field on this card. For example, HHH ^^^^^^C is used to start assignment of undefined address in core. HHH^^^^^^H is used to assign in high-speed storage.

CON, NUM, ZON, ALF. These provide four types of constants. The A field is treated exactly like the A field of an instruction. The M and C fields specify a 10-digit constant, except with ALF only M is used. CON means treat the contents of M and C as digits or the undigits A,B,C,F,G,H. NUM and ZON mean to treat M and C as an alphabetic constant and take the numeric or zone word of this constant (expressed in machine code). ALF takes the M field and produces a constant zzzzznnnnn where zzzzz is the zone part and nnnnn is the numeric part. On all four constant operators the IR field is used to specify the sign.

PAT. Print availability table. This causes 50 lines to be printed. Each line has five entries:

Level L L Level L+100 Level L+150

The availability word for a level contains 40 bits in machine code, corresponding to the status of the word in each band with the following band assignments:

00 02 ... 18
20 22 ... 38
40 42 ... 58
60 62 ... 78

As an example, at the beginning of assembly PAT would produce

BGGGG99999	GGGGG99999	0000000000	GGGGG99999	GGGGG99999
GGGGG99999	GGGGG99999	0000010000	GGGGG99999	GGGGG99999

etc.

(locations 0001-4999 are available).

TYP, ON , OFF. Three control operators are provided in case a person wants to use a single assembly deck to produce several almost identical versions, without

going through the card-to-tape pass each time. TYP causes the computer to stop at assembly time and then RL should be set to 000000nnnn. The operators OFF and ON have a defined M address and are effective only if the equivalent of M is nnnn matching a previous TYP. They temporarily shut assembly off or on. The FORTRAM system program is coded for five types:

8001:SS80, I
8002:SS80, II
9000:SS90, card
9001:SS90, tape I
9002:SS90, II

Note: The line B8AH NEW100001 00000 causes some assembly into the B00A-B99F area of core storage. Other uses of the control op NEW will not be listed here.

Operating Instructions

Loading the assembler:

rC	G2	0400	000A
rA	F6	B000	B000

Loading the object program:

rC	G2	0500	000A
rA	F6	7800	7801

(The assembler is presently set up for an 8800 word drum.)

Mount QADAAD tape on unit 4, scratch tapes on units 3 and 5. If there is tape input, put it on unit 2. (The card-to-tape pass copies from cards and/or unit 2 to unit 3. If this phase is omitted, mount a good input tape on unit 3 and forget unit 2.) If flowcharting is desired, you will need scratch tapes on units 6 and 7 also.

As soon as the assembler has loaded itself, it stops. Depress RUN if the card-to-tape pass is to be used else depress M and RUN to skip this pass.

When the second pass has loaded, it stops, and the operator should depress RUN.

The computer should next stop with HLT. B9AH, but several other stops may occur.

Pass 1 (card-to-tape) stops:

0001 Comparison failure on card reader. The bad cards have been diverted to stacker 2. Reload them, depress RUN.

0002 Card reader off normal or empty. Fix, depress RUN.

0005 Error in writing tape. This is really unfortunate. If RUN is depressed the condition is ignored.

0006 Error in reading tape. If RUN is depressed reading is attempted again, in the opposite direction. Gain should be varied by the operator.

Pass 2 (assembly) stops:

1111 Line numbers on input tape not sequential, indicates a bad situation on the input tape (unit 3)

2222 Tape error. Depressing RUN will cause the last tape instruction (read or write) to be tried again, with no backspacing provided. The last tape instruction appears in RA, and it is usually a G2 0300. In this case, restart by typing G2 0305 000A into rC, depress general clear and RUN.

3333 Printer off normal. Fix, depress RUN.

4444 Tape 3 not yet ready. Depress RUN.

8888 Parity error on tape buffer unload; depress RUN to ignore.

000A END card with undefined M address. Key into rA an address.

000B TYP card. Key into rL the type number, depress RUN.

1212 Normal stop for FLO. If flowcharting is to be bypassed, clear rA and RUN, else just RUN.

B9AH Normal stop at end of assembly. Depress M and RUN to continue assembly of another program. Otherwise, servo 5 contains a self-loading tape. Depress RUN to load it, or if flowcharting is used, to bring in pass 3.

Pass 3 (flowcharting) stops:

A stop occurs after pass 3 has loaded itself; merely depress RUN.

1111 Tape reading error. Depressing RUN will try to read again, but depending on the tape error you may wish to execute a read backwards. RX contains the tape instruction, so type in G2 0605 000C or G2 0705 000C.

3333 Printer off normal. Fix, depress RUN.

4444 Parity error on tape buffer unload; depress RUN to ignore.

5555 Invalid flow chart. Depress RUN until program starts up again.

At the completion of Pass 3 a stop will occur. If you wish to load the object program depress RUN. Depress M and RUN to continue flowcharting of another program if you are doing multiple assemblies.

This list does not include stops encountered while QADAAD is loading itself from tape.

Errors on assembly listing

On the assembly listing any error indications will appear to the left of the line number. On the END card the error indication will be blank only if there were no errors detected earlier.

Errors are identified by the number of the field where the error occurred.

Number:	1	2	3	4	5	6	7	8
Field:	A	AH	M	MH	C	CH	OP	DK

There are two special error indications which are perhaps only apparent errors:

-indicates the latency time has come out pretty badly. This may or may not be significant.

;indicates the H field instructions cannot be carried out; the next best alternative was tried. If the whole drum is unavailable 0000 is assembled.

The Automatic Documentation Feature

If automatic flowcharting is desired, the control operation FLO (not mentioned above) should appear at the head of the program. As soon as FLO appears, the remarks field of the listing is examined and has special significance. Three listings rathen than one are then produced:

I. The assembly listing
II. The algorithm listing
III. The flow chart

Listings II and III are produced simultaneously as a second pass to the assembly.

The documentation is presented in a fixed format which has been designed for effective description. When flowcharting, the program is broken into logical segments called sections. For example, a subroutine constitutes a section. Each section is given a single alphabetic letter to identify it; thus there is section A, section B, etc. One flowchart is prepared for each section.

In each section there are at most 99 subsections, each of which corresponds to a box on the chart. In section A, the subsections are A1, A2, ..., A10, A11, etc. In each box of the flowchart appears the subsection number and a few "key words" which tell generally what is taking place. The key-words are usually rather vague until one is familiar with the algorithm listing which tells specifically what is going on. (The algorithm listing should if possible also tell why it is going on.)

The remarks field is split up between the listings, as follows:

I The assembly listing contains the section names and with each subsection number the key words; and also special coding-oriented details.

II The algorithm listing contains the section names and with each subsection number the specific algorithm descriptions.

III The flow chart contains key words, condition branch names, appropriate boxes and connecting lines, and also line numbers from the assembly listing for cross reference.

The DK field

The first four columns of the remarks (cols 32-35) are of prime importance when flowcharting, and are called the DK or documentation key field. The following forms are used in DK fields:

1. G$\wedge\wedge\wedge$ Ordinarily the remarks are deleted from the assembly listing except section names and keywords, but G causes them to appear on the assembly listing, and not on the other listings. This is used to give coding-oriented details.

2. $\wedge\wedge\wedge\wedge$ No special operation. The remainder of the remarks are part of the algorithm listing only.

3. X$\wedge\wedge\wedge$ Same as $\wedge\wedge\wedge\wedge$ except this line is omitted from the assembly listing. This is used when remarks take more room than the machine language.

4. K.$\wedge\wedge$ Here K is the section letter. This line is the beginning of a new section, and causes a skip to next page on each listing.

5. Kn$\wedge$ or Knn. Indicates a new subsection. The remarks field columns 36-56 contains the key words.

6. CODI These are the first four letters of "Coding Details" which is part of the format explained below.

7. TABL This indicates this is a title section, with no subsections, and it is part of the format explained below.

8. Anything else is a condition name, e.g. YES: or NO: This means

a) If cols 36-40 are blank, this condition is to label the branch to the next box below.

b) Otherwise this condition branches to the next name appearing in the rest of the remarks.

Within the remarks of the algorithm description, branches to any but the next subsection are indicated by prefixing the name by the symbol # . This symbol is deleted from the final listing. If a condition name preceded, it is a conditional branch to this place. If no condition name preceded on this subsection it is an unconditional branch.

These rules are best explained by example, and so a listing of a small sample program is attached to this report.

Format for algorithm listing:

```
      P.  NAME OF PROGRAM (used for long programs only)
      TABLE OF CONTENTS
         A. NAME OF SECTION A
         B.  NAME OF SECTION B
         DESCRIPTION OF THE PROGRAM
   A.  NAME OF SECTION A
         DETAILS ABOUT THE PURPOSE OF THIS SECTION
         AND GENERAL SUMMARY.
  A1.  FIRST STEP
         WHAT HAPPENS AS THE FIRST STEP
  A3.  NEXT STEP
         WHAT HAPPENS AS THE NEXT STEP.  SUBSECTION
         NUMBERS MUST APPEAR IN ASCENDING ORDER, BUT
         NOT NECESSARILY SEQUENTIALLY
```

CODING DETAILS. AN OPTIONAL PART AT THE CLOSE OF A SECTION TELLS, FOR EXAMPLE, WHAT REGISTERS CONTAIN WHICH INPUTS AND OUTPUTS TO THIS SUBROUTINE.

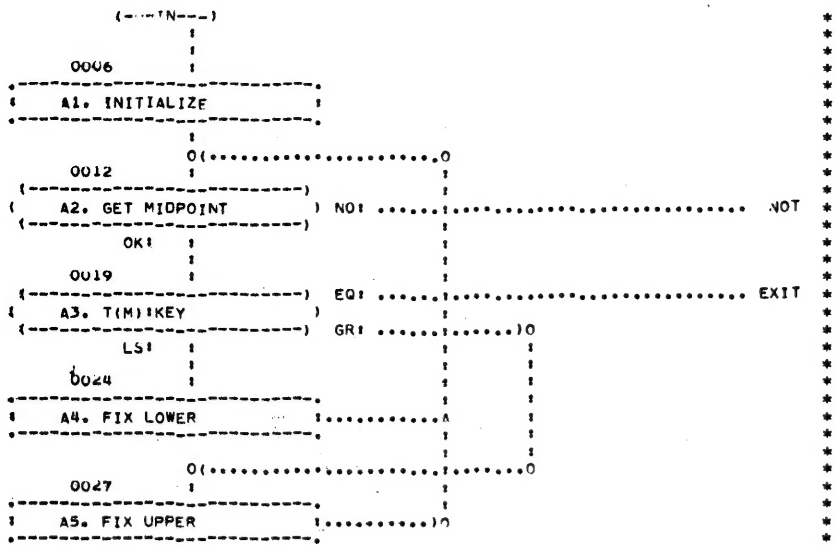
INPUT AS PUNCHED ON CARDS

```
         FLO                                A.  SERCH.
T0001    BLR  1000      1999                    THIS SUBROUTINE SEARCHES THROUGH TABLE T
LOWER    EQU  B01A                              TO SEE IF IT CAN FIND AN ENTRY MATCHING
UPPER    EQU  B02A                              A GIVEN KEY.
KEY      EQU  B03A
         HHH       H
SERCH    STA KEY                            A1. INITIALIZE
         LDA#00000      10000                   START OUT BY SETTING 'LOWER' TO 1,
         STA LOWER                              'UPPER'TO 1000.
         SHL  0300      1F                      THE TABLE IS T0001 THROUGH T1000 AND IS IN
1        STA UPPER                              ASCENDING SEQUENCE.
         ADD LOWER      3F
3        ATL                                A2. GET MIDPOINT
         MUL#00000      000A5                   SET 'M' TO (LOWER+UPPER)/2.  'M' WILL THUS
         LDX RA                                 APPROXIMATE THE MIDPOINT OF THE INTERVAL
         LDL LOWER                              WHERE WE HAVE PINPOINTED THE SEARCH.
         TGR 2F                             NO: IF 'UPPER' IS LESS THAN 'LOWER', THE KEY
         TEQ 2F         NOT                     IS#NOT IN THE TABLE.
2        ADD            RA                  OK:
         LDA T0000                          A3. T(M):KEY
         LDL KEY                                COMPARE T(M) WITH THE SEARCH KEY.
         TEQ2 0000                          EQ: IF EQUAL, WE#EXIT.
         TGR 2F                             GR: IF GREATER, TO#A5.
         LDA RX                             LS:
         ADD#00000      10000               A4. FIX LOWER
         STA LOWER                              SET 'LOWER' TO M+1,AS T(M) IS TOO SMALL.
         ADD UPPER      3B                      TO#A2.
2        LDA RX                             A5. FIX UPPER
         SUB            1B                      SET 'UPPER' TO M-1, AS T(M) IS TOO BIG.
         CON 00000      10000                   TO#A2.
                                            CODING DETAILS:  AT ENTRY RB2 CONTAINS THE EXIT
                                            X   LOCATION AND RA CONTAINS THE KEYWORD.
                                            X   IF FOUND, THE PLACE FOUND IS IN RX.
         HHH                                    IF NOT IN TABLE, EXIT OCCURS TO LOCATION'NOT'
NOT      HLT            *                   T.  TEST.
TEST     LIR1 0000      -T                  T1. SET UP T
-T       IIR1 0001                              FILL TABLE T, PUTTING 2I IN T(I).
         ADD RA
         STA1T0000
         IIR1 0000
         ADD            -T
         CON 99900      00000
&T       LDA#00010      00000               T2. SERCH 100.
         LIR2           SERCH                   USE THE SEARCH ROUTINE TO SEE IF 100 IS IN.
         ADD#00000      10000               T3. SERCH 101.
         LIR2&T         SERCH                   SEARCH ALSO FOR 101 WHICH IS#NOT IN THE TABLE
         END TEST
         FIN
```

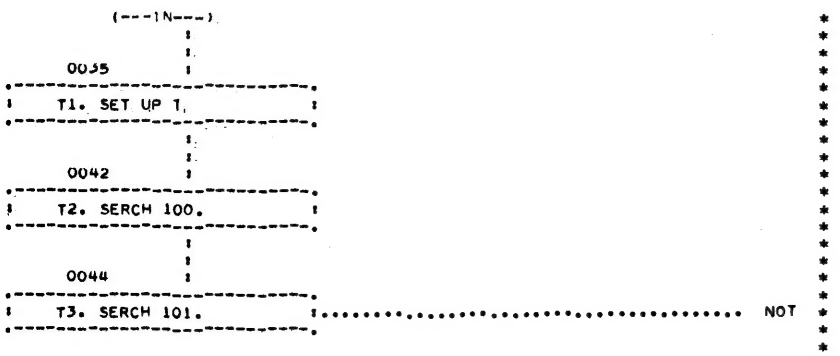
PAGE 1 OF OUTPUT

```
0000                                                  FLO                       A.  SERCH.
0001                                         T0001    BLR    1000       1999
0002                                         LOWER    EQU    B01A
0003                                         UPPER    EQU    B02A
0004                                         KEY      EQU    B03A
0005                                                  HHH            H
0006   4006  BBB 0 60  B03A  4010            SERCH    STA    KEY                A1. INITIALIZE
0007   4010  BBB 0 25  4012  4014                     LDA#   00000      10000
0008   4014  BBB 0 60  B01A  4018                     STA    LOWER
0009   4018  BBB 0 37  0300  4024                     SHL    0300       1F
0010   4024  BBB 0 60  B02A  4028            1        STA    UPPER
0011   4028  BBB 0 70  B01A  4033                     ADD    LOWER      3F
0012   4033  BBB 0 77  4033  4036            3        ATL                       A2. GET MIDPOINT
0013   4036  BBB 0 85  4038  4015                     MUL#   00000      000A5
0014   4015  BBB 0 05  000A  4019                     LDX    RA
0015   4019  BBB 0 30  B01A  4023                     LDL    LOWER
0016   4023  BBB 0 87  4026  4226                     TGR    2F
0017   4226  BBB 0 82  4026  4029                     TEQ    2F         NOT
0018   4026  BBB 0 70  4228  000A            2        ADD               RA
0019   4228  BBB 0 25  0999  4001                     LDA    T0000              A3. T(M):KEY
0020   4001  BBB 0 30  B03A  4005                     LDL    KEY
0021   4005  BBB 1 82  0000  4009                     TEQ2   0000
0022   4009  BBB 0 87  4212  4412                     TGR    2F
0023   4412  BBB 0 25  000C  4016                     LDA    RX
0024   4016  BBB 0 70  4218  4021                     ADD#   00000      10000   A4. FIX LOWER
0025   4021  BBB 0 60  B01A  4025                     STA    LOWER
0026   4025  BBB 0 70  B02A  4033                     ADD    UPPER      3B
0027   4212  BBB 0 25  000C  4216            2        LDA    RX                 A5. FIX UPPER
0028   4216  BBB 0 75  4418  4024                     SUB               1B
0029   4418  BBB 0 00  0001  0000                     CON    00000      10000
0030
0033                                                  HHH
```

```
0034  4029  BBB 0 67  4029  4029   NOT     HLT              *        T.  TEST.
0035  0035  BBB 0 08  0000  0038   TEST    LIR1  0000       -T       T1. SET UP T
0036  0038  BBB 0 0G  0001  0042   -T      IIR1  0001
0037  0042  BBB 0 70  000A  0047           ADD   RA
0038  0047  BBB 0 64  0999  0001           STA1  T0000
0039  0001  BBB 0 0G  0000  0005           IIR1   0000
0040  0005  BBB 0 70  0007  0038           ADD                    -T
0041  0007  BBB 0 99  9000  0000           CON   99900            00000
0042  0039  BBB 0 25  0041  0043   &T      LDA#  00010            00000    T2. SERCH 100.
0043  0043  BBB 1 02  0247  4006           LIR2                   SERCH
0044  0247  BBB 0 70  0049  0052           ADD#  00000            10000    T3. SERCH 101.
0045  0052  BBB 1 02  0039  4006           LIR2  &T               SERCH
0046                                       END   TEST
```




```
*   A.  SERCH.
*       THIS SUBROUTINE SEARCHES THROUGH TABLE T
*       TO SEE IF IT CAN FIND AN ENTRY MATCHING
*       A GIVEN KEY.
*   A1. INITIALIZE
*       START OUT BY SETTING 'LOWER' TO 1.
*       'UPPER'TO 1000.
*       THE TABLE IS T0001 THROUGH T1000 AND IS IN
*       ASCENDING SEQUENCE.
*   A2. GET MIDPOINT
*       SET 'M' TO (LOWER+UPPER)/2.  'M' WILL THUS
*       APPROXIMATE THE MIDPOINT OF THE INTERVAL
*       WHERE WE HAVE PINPOINTED THE SEARCH.
*       IF 'UPPER' IS LESS THAN 'LOWER', THE KEY
*       IS NOT IN THE TABLE.
*   A3. T(M)!KEY
*       COMPARE T(M) WITH THE SEARCH KEY.
*       IF EQUAL, WE EXIT.
*       IF GREATER, TO A5.
*   A4. FIX LOWER
*       SET 'LOWER' TO M+1.AS T(M) IS TOO SMALL.
*       TO A2.
*   A5. FIX UPPER
*       SET 'UPPER' TO M-1. AS T(M) IS TOO BIG.
*       TO A2.
*   CODING DETAILS:  AT ENTRY RB2 CONTAINS THE EXIT
*       LOCATION AND RA CONTAINS THE KEYWORD.
*       IF FOUND, THE PLACE FOUND IS IN RX.
*       IF NOT IN TABLE, EXIT OCCURS TO LOCATION'NOT'
*
*
*
```




```
*   T.  TEST.
*   T1. SET UP T
*       FILL TABLE T. PUTTING 2I IN T(I).
*   T2. SERCH 100.
*       USE THE SEARCH ROUTINE TO SEE IF 100 IS IN.
*   T3. SERCH 101.
*       SEARCH ALSO FOR 101 WHICH IS NOT IN THE TABLE
```

OUTPUT OF SYMBOL-USAGE ROUTINE

-T	0036	0035	0040		
&T	0042	0045			
KEY	0004	0006	0020		
LOWER	0002	0008	0011	0015	0025
NOT	0034	0017			
RA	0014	0013	0037		
RX	0023	0027			
SERCH	0006	0043	0045		
TOOOO	0001	0019	0038		
TEST	0035	0046			